1. **Khái niệm về cơ sở dữ liệu: Phân loại cơ sở dữ liệu: SQL, NoSQL. Mục đích sử dụng của CSDL**

- Khái niệm: Cơ sở dữ liệu (Database) là một tập hợp các dữ liệu có tổ chức và được lưu trữ trong một hệ thống máy tính. Nó cung cấp một cơ chế để lưu trữ, truy xuất, cập nhật và quản lý dữ liệu một cách hiệu quả.
- Phân loại:
    1. SQL: Cơ sở dữ liệu quan hệ (SQL) là một hệ thống quản lý cơ sở dữ liệu dựa trên mô hình quan hệ. Trong SQL, dữ liệu được tổ chức thành các bảng có mối quan hệ với nhau thông qua các khóa chính và các khóa ngoại. SQL sử dụng ngôn ngữ truy vấn cấu trúc (Structured Query Language) để truy xuất và quản lý dữ liệu. Một số hệ quản trị cơ sở dữ liệu quan hệ phổ biến là MySQL, Oracle, SQL Server và PostgreSQL. SQL thường được sử dụng trong các ứng dụng có dữ liệu có cấu trúc, yêu cầu tính nhất quán và đảm bảo tính toàn vẹn của dữ liệu.
    2. NoSQL: Cơ sở dữ liệu phi quan hệ (NoSQL) là một hệ thống quản lý cơ sở dữ liệu không dựa trên mô hình quan hệ. NoSQL không yêu cầu cấu trúc dữ liệu cố định và cho phép lưu trữ dữ liệu phi cấu trúc hoặc có cấu trúc linh hoạt hơn. NoSQL hỗ trợ việc lưu trữ và truy xuất dữ liệu một cách linh hoạt và phù hợp với các mô hình dữ liệu không nhất quán hoặc có tính mở rộng cao. Các loại cơ sở dữ liệu NoSQL bao gồm cơ sở dữ liệu cột (columnar), cơ sở dữ liệu tài liệu (document-based), cơ sở dữ liệu key-value và cơ sở dữ liệu đồ thị (graph-based). Một số hệ quản trị cơ sở dữ liệu NoSQL phổ biến là MongoDB, Cassandra, Redis và Neo4j.
- Mục đích:
    1. Lưu trữ dữ liệu: CSDL được sử dụng để lưu trữ dữ liệu một cách có tổ chức. Dữ liệu có thể bao gồm thông tin về khách hàng, sản phẩm, giao dịch, dữ liệu kho, thông tin địa lý và nhiều loại dữ liệu khác. Bằng cách lưu trữ dữ liệu trong CSDL, người dùng có thể dễ dàng truy cập, tìm kiếm và cập nhật dữ liệu theo nhu cầu.
    2. Quản lý dữ liệu: CSDL cung cấp các công cụ và chức năng để quản lý dữ liệu. Nó bao gồm việc tạo, xóa, cập nhật và sao lưu dữ liệu. Người quản trị cơ sở dữ liệu có thể sử dụng CSDL để kiểm soát quyền truy cập vào dữ liệu, duy trì tính toàn vẹn và bảo mật của dữ liệu.
    3. Truy vấn và phân tích dữ liệu: CSDL cho phép thực hiện các truy vấn để truy xuất thông tin từ dữ liệu. Người dùng có thể sử dụng

ngôn ngữ truy vấn (như SQL) để tạo các câu truy vấn phức tạp để tìm kiếm, lọc và tính toán dữ liệu. Điều này hỗ trợ trong việc phân tích dữ liệu, tìm hiểu xu hướng, tạo báo cáo và đưa ra quyết định dựa trên dữ liệu.

4. Tích hợp và chia sẻ dữ liệu: CSDL cho phép nhiều ứng dụng và hệ thống khác nhau truy cập và chia sẻ dữ liệu. Bằng cách sử dụng giao thức và công nghệ tương thích, CSDL có thể tích hợp với các ứng dụng khác nhau, cho phép chúng truy cập và sử dụng dữ liệu chung một cách nhất quán.
5. Tăng hiệu suất và độ tin cậy: CSDL cung cấp các công cụ và kỹ thuật để tăng hiệu suất và độ tin cậy của hệ thống. Điều này bao gồm việc tối ưu hóa truy vấn, quản lý tài nguyên, đảm bảo sự nhất quán và đồng bộ dữ liệu, xửlý lỗi và khôi phục dữ liệu sau sự cố.

Tóm lại, CSDL được sử dụng để lưu trữ, quản lý và cung cấp truy cập vào dữ liệu một cách hiệu quả, giúp hỗ trợ các ứng dụng phức tạp, phân tích dữ liệu và quản lý dữ liệu một cách tin cậy.

2. Tìm hiểu 1 trình quản lý SQL bất kỳ: postgresql

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được phát triển từ nguồn gốc của hệ thống quản lý cơ sở dữ liệu Ingres. Nó được coi là một trong những hệ quản trị cơ sở dữ liệu quan hệ phổ biến nhất và mạnh mẽ nhất.

Một số điểm nổi bật về PostgreSQL:

1. Tính năng và độ tin cậy: PostgreSQL cung cấp nhiều tính năng phong phú và đáng tin cậy. Nó hỗ trợ các tính năng quan trọng như kiểm soát phiên bản, giao dịch, khóa chính và khóa ngoại, truy vấn phức tạp, chức năng và các loại dữ liệu tùy chỉnh. Nó cũng hỗ trợ các tính năng mở rộng như truy vấn địa lý và phân tích dữ liệu.
2. Mở rộng và hiệu suất: PostgreSQL có khả năng mở rộng tốt, cho phép xử lý dữ liệu lớn và tải công việc cao. Nó hỗ trợ các cơ chế như phân vùng, sao lưu, khôi phục, nhân bản và cân bằng tải để đảm bảo hiệu suất cao và tính sẵn sàng của hệ thống.
3. Bảo mật: PostgreSQL có nhiều tính năng bảo mật để bảo vệ dữ liệu. Nó hỗ trợ xác thực người dùng, quản lý quyền truy cập cơ sở dữ liệu và mã hóa dữ liệu. Nó cũng có khả năng kiểm tra bảo mật và giám sát để giám sát và kiểm soát truy cập vào cơ sở dữ liệu.

4. Hỗ trợ đa nền tảng: PostgreSQL có sẵn cho nhiều hệ điều hành như Linux, Windows và macOS. Điều này cho phép nó hoạt động trên nhiều môi trường và tích hợp với nhiều ứng dụng khác.
5. Cộng đồng và tài liệu phong phú: PostgreSQL có một cộng đồng rộng lớn và nhiệt tình, với nhiều tài liệu hướng dẫn, tài liệu tham khảo và diễn đàn trực tuyến. Người dùng có thể tìm kiếm hỗ trợ và chia sẻ kiến thức với cộng đồng PostgreSQL.
6. Mã nguồn mở: PostgreSQL là một phần mềm mã nguồn mở, điều này có nghĩa là mã nguồn của nó có sẵn và có thể được xem xét, chỉnh sửa và phát triển bởi cộng đồng. Điều này mang lại linh hoạt và khả năng tùy chỉnh cho người dùng.

3. Tìm hiểu các cú pháp cơ bản của SQL: CREATE TABLE, INSERT INTO , SELECT, FROM, DELETE, UPDATE

- **CREATE TABLE**

  CREATE TABLE table_name (column_name1 data_type(size), column_name2 data_type(size),column_name3 data_type(size), ...);

- **INSERT INTO**

  INSERT INTO table_name (column1,column2,column3)

  VALUES (value1,value2,value3);

- **SELECT, FROM**

  SELECT name_column1, name_column2 **FROM** name_table;

- **DELETE**

  DELETE FROM table_name **WHERE** condition;

- **UPDATE**

  UPDATE table_name
  SET column1 = value1, column2 = value2, ...
  WHERE condition;

4. Kết nối java với sql: jdbc

Các bước giao tiếp giữa chương trình Java và Database:

- Load Driver.
- Tạo kết nối (Open Connection).
- Tạo câu lệnh truy vấn SQL (Statement).
- Thực thi câu lệnh truy vấn SQL (Execute Query).
- Đóng kết nối (Close Connection).

Load Driver

Để kết nối với database, chúng ta cần load driver và register nó với ứng dụng. Có 2 cách để thực hiện:

Class.forName() : load class Driver trong memory tại thời điểm runtime. Để đăng ký gọi phương thức: **Class.forName("driverName");**

DriverManager.registerDriver() : DriverManager là class có sẵn trong Java. Để đăng ký gọi phương thức: **DriverManager.registerDriver(new DriverName());**

Tạo kết nối (Open Connection)

Sau khi đã load Driver, chúng ta tạo connection:

```
DriverManager.getConnection(connectionURL); // Or
DriverManager.getConnection(connectionURL, userName, password);
// Pattern String connectionURL =
"jdbc:mysql://hostname:port/dbname";
// Example String connectionURL =
"jdbc:mysql://hostname:3306/jdbcdemo";
Connection con = DriverManager.getConnection(connectionURL,
username, password);
```

Tạo câu lệnh truy vấn SQL (Statement)

Sau khi một kết nối được thiết lập, chúng ta có thể tương tác với cơ sở dữ liệu.

Các interface JDBCStatement, CallableStatement và PreparedStatement xác định các phương thức cho phép gửi các lệnh SQL và nhận dữ liệu từ cơ sở dữ liệu.

```
Statement st = con.createStatement();
```

Thực thi câu lệnh truy vấn SQL (Execute Query)

Bây giờ đến phần quan trọng nhất, đó là thực hiện câu lệnh truy vấn SQL (execute query). Như chúng ta biết, có thể có nhiều loại truy vấn. Một trong số đó là:

- Truy vấn để cập nhật (update) / chèn (insert)/ xóa (delete) trong cơ sở dữ liệu.
- Truy vấn để lấy dữ liệu (select).
- Statement cung cấp một số phương thức để thực thi truy vấn SQL tương ứng với các loại trên:
- Phương thức **execQuery()** : được sử dụng để thực hiện các truy vấn truy xuất giá trị từ cơ sở dữ liệu (select). Phương thức này trả về đối tượng ResultSet có thể được sử dụng để lấy tất cả các dữ liệu (record) của bảng.
- Phương thức **execUpdate()** : được sử dụng để thực hiện các truy vấn insert/ update/ delete.
- Phương thức **execute()** : có thể thực thi cả 2 trường hợp trên. Nếu phương thức statement.getUpdateCount() trả về số lượng record bị affect.

    - Nếu giá trị **> 0**, có nghĩa là thực thi các câu lệnh insert/ update/ delete.
    - Nếu giá trị **= 0**, có nghĩa là thực thi các câu lệnh insert/ update/ delete không có dòng nào bị ảnh hưởng hoặc thực thi câu lệnh cập nhật data structure.
    - Nếu giá trị **= -1**, có nghĩa là thực thi câu lệnh select. Khi đó, có thể gọi tiếp lệnh statement.getResultSet() để lấy ResultSet.

Đóng kết nối (Close Connection)

Cuối cùng, sau khi đã sử dụng chúng ta cần phải gọi phương thức **close()** để đóng kết nối (Connection) và giải phóng tài nguyên.

Bằng cách đóng kết nối, các đối tượng của Statement và ResultSet sẽ được đóng tự động. Tuy nhiên, chúng ta nên tập thói quen close() Statement sau khi sử dụng hay vì chờ đợi điều đó xảy ra khi nó tự động bị đóng để giải phóng tài nguyên. Đặc biệt nếu chúng ta thực thi Statement trong vòng lặp, thì có thể sẽ gặp vấn đề về thiếu tài nguyên sử dụng nếu chờ đợi nó tự động đóng.

Khi Statement được close() thì ResultSet của Statement cũng được close.